Artemisa: an Open-Source Honeypot Back-end to Support Security in VoIP Domains

Rodrigo Do Carmo, Mohamed Nassar, Olivier Festor

Artemisa: an Open-Source Honeypot Back-End to Support Security in VoIP Domains

Rodrigo do Carmo
Blas Pascal University
Av. Donato Álvarez 380
5147 Argüello, Córdoba, Argentina
Email: rdocarmo@ubp.edu.ar

Mohamed Nassar
INRIA Research Center
Nancy - Grand Est
615, rue du jardin botanique
54602 Villers-Lès-Nancy, France
Email: nassar@loria.fr

Olivier Festor
INRIA Research Center
Nancy - Grand Est
615, rue du jardin botanique
54602 Villers-Lès-Nancy, France
Email: festor@loria.fr

***Abstract*—Voice over IP (VoIP) and the Session Initiation Protocol (SIP) are establishing themselves as strong players in the field of multimedia communications over IP, leveraged by low cost services and easy management. Nevertheless, the security aspects are not yet fully mastered. In this paper we present an open-source implementation of a VoIP SIP-specific honeypot named Artemisa. The honeypot is designed to connect to a VoIP enterprise domain as a back-end user-agent in order to detect malicious activity at an early stage. Moreover, the honeypot can play a role in the real-time adjustment of the security policies of the enterprise domain where it is deployed. We aim, by this contribution, to encourage the deployment of such honeypots at large scale and the collection of attack traces. We test the capacity of the honeypot to handle a series of known SIP attacks and present results from diverse scenarios.**

I. Introduction

Voice over IP (VoIP) is quickly hitting the market after the establishment of the Session Initiation Protocol (SIP [9]) as the de-facto standard signaling protocol in the Internet and the IP Multimedia Subsystems (IMS). Basically, SIP allows two communicating parties to establish, modify and terminate a media session. The media session is described through the Session Description Protocol (SDP) body carried by the SIP messages. SIP is a request-response layered protocol: a SIP Dialog is composed of one or more transactions. A transaction is formed by a request, one or more informational responses and a final response. There are four types of transactions: on the one hand client transactions and server transactions, on the other hand INVITE transactions and non-INVITE transactions. The SIP addressing scheme is based on the Uniform Resource Identifier (URI) scheme like `sip:user@host:port;parameters`. A SIP enterprise domain is typically composed of user premises (hard and soft-phones), a SIP infrastructure (e.g., proxy, registrar, back-to-back user-agent) and supporting services (e.g., web-based management, TFTP, DNS).

VoIP has not only the inherited security problems of the network layers, but also new threats and vulnerabilities. The unprotected VoIP products represent new opportunities for the attackers to bypass firewalls and network security policies. The main IP telephony threats are: request flooding and Denial of Service (DoS) affecting the service availability, eavesdropping and privacy unveiling, media injection and alteration such as man-in-the-middle attacks, toll fraud and service theft, impersonation and call redirection to a malicious party, caller-ID spoofing, unsolicited call or what is known as SPAM over Internet Telephony (SPIT) [2]. The future of the VoIP security is hard to be predicted today, mostly because of the lack of information about security incidents and exploits.

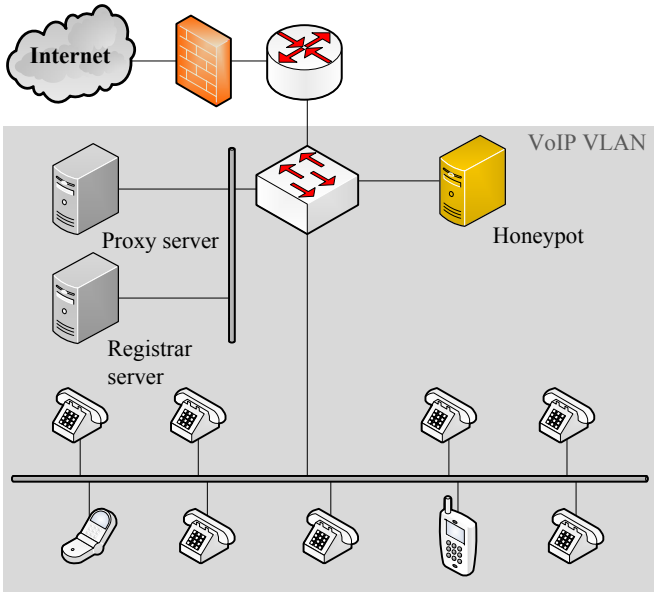

Figure 1. VoIP honeypot deployment

In front of these threats, innovative security approaches should be extended to cover this arena. In particular, the information gathering and the early warning systems are widely deployed in computer networks today as an important security component. The honeypots and the honeynets constitute an important factor in order to know what are the real dangers and what the attackers methodologies will be [13]. We do adapt this concept to the VoIP sphere.

In this context, we propose an architectural design together with an open-source implementation of a VoIP SIP-specific honeypot that can be deployed as a user-agent back-end in a

978-1-4244-9220-6/11/$26.00 ©2011 IEEE

VoIP enterprise domain. Because of the fact that the honeypot extensions do not represent real users, every activity targeting them is perceived as suspicious. The honeypot answers the calls and records them along with the SIP trace. The honeypot is able to classify many kinds of anomalies and report them to the administrator or automatically control the security policy of the domain under protection. The typical honeypot deployment configuration is depicted in Figure 1: the honeypot registers to one or several SIP registrars and waits for calls and SIP messages. Two options are possible: The first option is to connect the honeypot in the demilitarized zone. Thus it will be isolated from the internal network and the VoIP VLAN in case the machine running it is compromised. The second option is to connect it through the Internet as many VoIP providers today has subscribers from all over the globe across the Internet.

The rest of this paper is composed as follows: In Section 2 we expose related work on VoIP security and honeypots. Section 3 describes the functional goals of our design. In Section 4 we expand the software architecture and expose implementation details. Experimentation and results are shown in Section 5. Finally, Section 6 concludes the paper and presents the future works.

II. Related work

VoIP security has stimulated the enterprises, research and government communities to investigate the best ways of assuring its different security aspects. General recommendations for a safe VoIP deployment are defined in [14], [1], [5]. Several books have already been published on this topic [12]. Geneiatakis et al. [4] analyzed the SIP protocol and highlighted its weaknesses. SIP product vulnerabilities have also been notified. Sengar et al. [11] presented a threat model for integrated signaling between VoIP and PSTN and proposed an integral protection solution. The VoIP security clearly imposes new challenges because of its dynamic, open and large-scale settings. Therefore, innovative and novel defense approaches are required.

One of the issues that is extensively studied is the SPIT problem. The key issue with SPIT identification is the caller identity. Quittek et al. [8] applied hidden Turing tests on the caller side and compared their results to typical human communication patterns. For passing these tests, significant resource consumptions at the SPIT generating side were required, which contradicts the objective of the spammer of placing as many SPIT calls as possible. Quinten et al. [7] gave a survey of protection techniques against SPIT. VoIP SEAL [10] implements a two-stage decision process: The first stage contains modules that analyze a call only by looking at information which is available before actually answering the call. The second stage consists of modules that interact with the caller or the callee during the call. Since the second stage modules introduce some inconvenience, a scoring system is deployed at the first stage to determine if there is need to use them or not. In addition to the Turing test module, VoIP SEAL includes white- and blacklisting, simultaneous calls, call rate, and URI, IP, and domain correlation. The end-user feedback is taken into account for the SIP clients that are instrumented with this capability. Our approach in this paper is a complementary system that can interact with the existing solutions.

There are many projects deploying VoIP honeypots based on available VoIP software (such as Sipp[1]): The Honeynet Project[2] deploys several low interaction sensors providing trivial functionalities. They do not interact, or trick the attacker into making calls or else, but simply log connection attempts[3]. Particularly, the Australian and the Norwegian chapters announced several events[4] where the employment of the SIPVicious[5] tool is noticed. SIPVicious allows scanning for open SIP ports, enumerating user accounts, determining those that can be registered without authentication and cracking weak passwords by dictionary attacks.

Nassar et al. [6] have proposed a VoIP honeypot architecture equipped with reconnaissance tools and an inference engine —based on Bayes inference and Markov chains— that determines the nature of a received SIP message. This probabilistic approach cannot be applied without sufficient labeled data for training. A rule-based approach can be used instead. In this paper we extend this work and provide a real implementation of a honeypot within realistic VoIP settings.

Regarding the announced security events, one concludes that malicious attempts are in their infancy. There are many "hacking" tools, however, that can be taken "off-the-shelf" and used by script kiddies. We aim by this contribution to encourage the deployment of VoIP honeypots on large and early scales before the proliferation of large-scale attacks.

III. Functional goals

The honeypot registers several series of virtual extensions at one or more SIP registrars. The virtual extensions have to be chosen in order to protect the real ones. For example if the real extensions are all composed of three digits, we recommend that the virtual extensions cover all the 3-digit numbers that are used by real subscribers. Alternatively, the domain proxy or PBX can be configured to forward all the messages that are not addressed to real extensions towards the honeypot. The functional goals of such a setting range from enumeration detection and VoIP Spam mitigation to signature collection and attackers blacklisting. The security policy of the VoIP domain can be controlled in real time. We expand each of these goals next.

A. Enumeration detection

An enumeration is detected as a series of OPTIONS, REGISTER or INVITE messages addressed to a series of virtual extensions. The attack source must be able to receive the responses in order to analyse them. In active mode, our honeypot gathers information about the source (e.g., the attack tool, the user-agent, the IP address and domain, the geographical location, the autonomous system number) and reports them to

[1] http://sipp.sourceforge.net/
[2] http://www.honeynet.org/
[3] http://honeynet.org.au/?q=phoneynet_part2
[4] http://www.honeynor.no/
[5] http://sipvicious.org/blog/

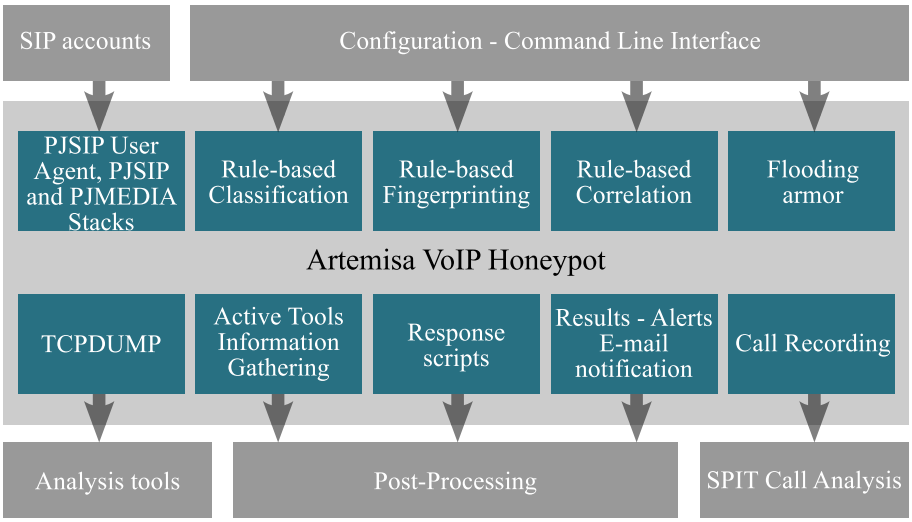

Figure 2. The honeypot modules

the domain administrator. The gathering of information is based on networking tools such as Nmap, Sipsak and DNS lookup. This information can be interpreted automatically in order to block the enumeration in real-time.

B. VoIP SPAM mitigation

Spam over Internet Telephony (SPIT) is detected as a series of INVITE messages addressed to a series of honeypot extensions followed by deliveries of media. The spitters wait for the 200 OK responses carrying the received SDP and send their spam to the announced IP and RTP port. Our honeypot responds to the SPIT calls and records them for further content-based analysis. The SPIT call analysis is helpful for deploying anti-spam filters within voice mail-boxes or for applying Turing tests for distinguishing the human from the automated SPIT patterns. The honeypot gathers pieces of information about the SPIT source and reports them to the domain administrator. These pieces can be interpreted automatically and used to block the spammers.

C. Signatures collection

Different VoIP equipments and firmwares (such as phones, call managers, servers) have been identified to be vulnerable to attacks such as remote crash, SQL injection, XSS cross-site scripting and remote eavesdropping. Theses vulnerabilities can be exploited by sending especially crafted SIP messages. A fingerprinting of the victim device may precede the attack in order to choose a suitable exploit. Our honeypot announces fake fingerprints (e.g., manufacturer string, product name, firmware version) in the user-agent or server header when it is actively fingerprinted (for example through an OPTIONS message). For example, a VoIP domain administrator may be interested by announcing similar fingerprints to what is used in the production domain. Likewise, the crafted SIP messages that are received will be collected and reported to the domain administrator. These messages are important in order to extract signatures and incorporate them in SIP-aware firewalls. We report the zero-days exploits to the manufacturers in order to patch their systems. We report and black-list the attack sources if they are identified to not being spoofed.

D. Real-time closed-loop control of the domain security policy

As aforementioned in the three previous functionalities, we aim to automate the interpretation and the reconnaissance of the perceived activities in real-time. The results of this analysis are reported to the administrator but can also trigger response scripts and exchange data with other components in the VoIP domain (e.g., SIP firewall, PBX dial-plan). The response can be applied in a progressive and continuous manner especially when the attacks persist. The closed-loop control of the openness of the domain permits to establish a compromise between the security and the reachability of the service. For example when dealing with SPIT, we can start by a “try later” policy. If the attack persists, we end by blocking totally the IP sources. From another side, the real time reaction to the security threats is mandatory for high-cost and sensitive services such as VoIP.

E. Configuring the honeypot

The honeypot functionalities are defined through a behavior mode represented by a state machine. The state machine is described using rules and fine-grained operations. Customized modes can be defined by the administrator. In addition, we define two primitive modes: passive and active. In passive mode, the honeypot does not use any networking tool hence does not send any request that may unveil its presence. In active mode and when the honeypot reveals that the attack source is real, it tries to collect information about it such as: the SIP fingerprint, the underlying operating system fingerprint, the IP route, the opened ports and services (several opened ports might indicate that many sessions are maintained in parallel). All these operations are supported by a modular and extensible architecture as described next.

IV. Honeypot modules and implementation

Artemisa is composed of several modules bundled together using the Python scripting language. These modules are depicted in Figure 2. The Tcpdump module controls the collection of the raw network traffic at the honeypot machine using the Tcpdump tool. The call recording module uses the PJMedia library to safeguard the received audio flows in appropriate formats. The remaining modules are briefly described next.

A. PJSIP User Agent, PJSIP and PJMEDIA stacks

The SIP User Agent is responsible of registering the virtual extensions at one or several registrars, answering calls, responding to SIP requests and announcing the virtual fingerprints. In addition, it controls the other components based on the behavior mode. We use the Python binding of the C/C++ PJSIP[6] User-Agent library (PJSUA). This library makes use of two stacks: PJSIP and PJMedia for SIP and RTP handling. PJSUA allows to emulate a SIP user agent in a very easy way. In contrast, the library binding does not offer full access to the SIP stack (e.g., for handling OPTIONS messages).

The PJSUA is supplied with the virtual SIP accounts and configured through setting files at bootstrap and a Command Line Interface (CLI) under execution.

B. Active tools - Information gathering

This component is responsible of running reconnaissance tools in order to collect complementary information about the source of received messages. We investigate several items in the SIP and SDP message: the IP source of the message, the caller URI, the Contact header, the Via header(s), the Route and Record-Route headers if any, the media IP and port (in the SDP attributes: `Connection(c)`, `Owner(o)` and `Media(m)`). This component invokes the following tools:

- Dig: being the alternative of the former Nslookup, this tool is used for querying the DNS records. It performs DNS and Reverse DNS lookups.
- jwhois: being an Internet Whois, this tool helps to obtain publicly available information about the domain names involved in the SIP message.
- Sipsak[7]: this is a SIP tool that can make different SIP tests such as sending an OPTIONS message to a target URI.
- Nmap[8]: this is an efficient scanning tool that we can use to scan the caller's host and determine if SIP, media, or both ports are open.
- Traceroute: helps trace the IP routes toward the different SIP proxies and UAs involved in the SIP message.
- p0f[9]: is a passive fingerprinting tool that helps us retrieve additional information about the source such as the operating system.

The collected information are formatted and supplied to the set of classification and correlation rules.

[6]http://www.pjsip.org/
[7]http://sipsak.org/
[8]http://nmap.org/
[9]http://lcamtuf.coredump.cx/p0f.shtml

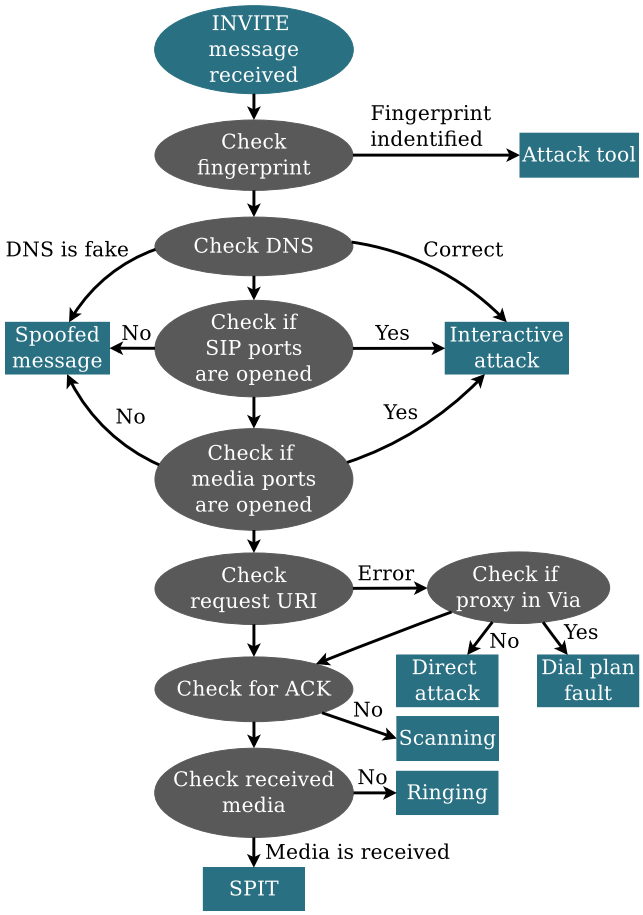

Figure 3. Decision tree for a received INVITE

C. Rule-based classification

The classifier interprets the data obtained by the reconnaissance tools and generates qualitative conclusions based on rules forming a decision tree. An example is shown in Figure 3. This decision tree interprets a received INVITE message and generates one or more conclusions about the nature of the message. The ovals represent actions performed and the rectangles (the leaf nodes) represent conclusions.

The first and most important step is to check if the message has any known fingerprint of an attack tool, since it is easy to deal with an attack if we know how it was generated.

The second step is to determine if the message is spoofed or not. Messages that are not spoofed is a sign of interactive attacks where the attacker needs to catch the responses and analyse them. This can be done by checking for the used domain names to be real, and checking if the announced SIP, media, or both ports are opened.

The third step is to check if the request URI carries one of the virtual extensions. If not, we check if the message is coming across the used proxy (Via header), or PBX (IP source). This can differ a direct attack (probably from inside the network) from a dial-plan error (why this message is routed to the honeypot while it does not address a virtual extension?).

Other checks apply to the overall dialog as initiated by this INVITE message. The honeypot answers with a 200 OK and waits for ACK and media. If received, the dialog is considered as SPIT. Otherwise, the INVITE needs to be correlated with other events because it could be part of a scanning (trying to reach a gateway to the PSTN or valid extensions) or a ringing attempt (ringing all the phones in the target domains).

We give the administrator the ability to define the investigation process operations, and the rules to be applied on the investigation results, hence to control the decision process of the honeypot. The classification rules are placed under a context (for example the received INVITE message) and are composed of a pre-action, a condition and an action. We use a simple `attribute = value` syntax to describe the rules. For example, the first node of the aforementioned decision tree can be represented by the following rule:

```
[rule_1]
type = Classification
context = INVITE
pre-action = assign %a Check_Fingerprint()
condition= %a in FINGERPRINTS
action= add \
"Tool identified as: %a" CONCLUSIONS
```

where `assign`, `in` and `add` are specific operators, `FINGERPRINTS` and `CONCLUSIONS` are pre-defined data arrays, and `Check_Fingerprint` is a pre-defined operation.

D. Rule-based fingerprinting

The fingerprinting rules are applied when the `Check_Fingerprint` operation is performed. A fingerprinting rule looks for a regular expression in a specified SIP header or attribute, or in the entire message. For example the following rule looks for the chain of characters `friendly-scanner` in the User Agent header of the SIP message in question:

```
[rule_2]
type = Fingerprinting
context = INVITE
re = 'friendly-scanner'
where = SipMessage.UserAgent
action = return "SIPVicious"
```

E. Rule-based correlation

The correlation rules are applied when several SIP messages are used to infer a conclusion. We define different types of correlation rules. These types are particularly needed to detect flooding, scanning and SPIT series of events.

This type of rules support timer and threshold attributes. For example, the following rule is used to check if an ACK is received after an INVITE in a time window of 5 seconds.

```
[rule_3]
type = Correlation.EventInWindow
context = INVITE
scope = DIALOG
timer = 5
condition = context == ACK
action-if-false = add \
"no ACK is received" CONCLUSIONS
```

The `scope` attribute defines to which SIP messages this rule can be applied. `scope = DIALOG` means that the rule can be solely applied to messages that have the same call-ID of the former INVITE. The `action-if-false` attribute means that the action will be executed in case the window time has expired and the condition above is still false (in contrast to the `action` attribute).

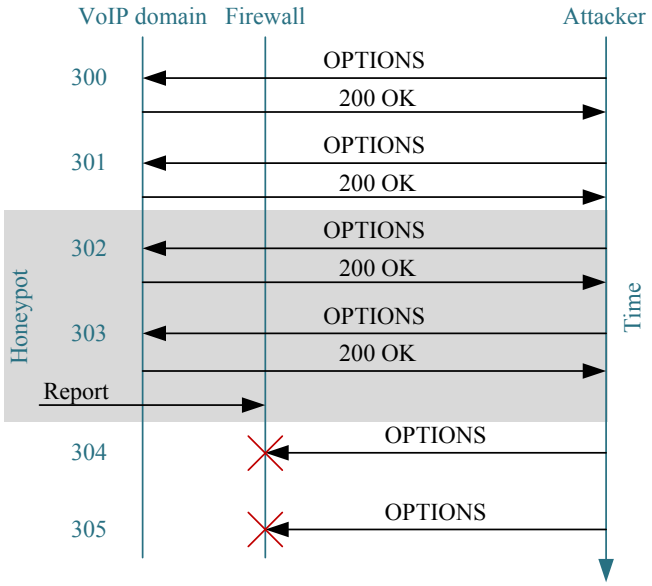

Figure 4. Real-time integration of the honeypot results

F. Flooding armor

This component protects the honeypot from been flooded by SIP requests. This is done by ensuring that the honeypot processes a limited number of requests in a given period of time. The correlation rules are used. For example, the following rule detects if three INVITE messages are received from the same IP source in a time window of one second. The required action is executed at the end.

```
[rule_4]
type = Correlation.ThresholdInWindow
context = INVITE
scope = IPSrc
timer = 1
threshold = 3
action = system "bash ./on_flood.sh %IPSrc"
```

G. Response scripts

The response scripts are executed by the honeypot in order to react to a detected attack by blocking or mitigating it. For example, Figure 4 shows an enumeration attempt that is blocked in real time after being detected by the honeypot. The response scripts are called within the rule definitions by using the `system` keyword and passing the necessary arguments about the attack source (e.g., IP source, SIP From URI). The reaction can be performed at different levels: IP firewall, SIP-aware firewall, dial-plan, among others. The administrator defines his appropriate scripts in respect to the domain settings.

```
*************** Information about the call *****************
From:  in 192.168.0.103:9/udp
To: 500 in 192.168.0.104
Contact:  in 192.168.0.103:9/udp
Connection: 192.168.0.103
Owner: 192.168.0.103
Via 0: 192.168.0.103:9/udp
User-Agent: Elite 1.0 Brcm Callctrl/1.5.1.0 MxSF/v.3.2.6.26
                              .
                              .
                              .
 ********************** Conclusions ************************
* Tool identified as: Inviteflood
* Spoofed Message
* no ACK is received
* no Media is received
* Scanning or Ringing or flooding attack
* waiting for correlation results

Alert is saved on file 2010-07-17_2.txt
Alert is saved on file 2010-07-17_5.html
E-mail notification is disabled.
```

Figure 5. Example of a message Alert

```
INVITE sip:00442075005000@x SIP/2.0
Via: SIP/2.0/UDP IP hidden:58585;branch=z9hG4bKaergjerugroijrgrg
To: <sip:x>
From: <sip:IP hidden:58585>;tag=Zerogij34
Call-ID: 213948958-34384780214-384748@IP hidden
CSeq: 1 INVITE
Max-Forwards: 69
Contact: <sip:sip@IP hidden:58585;transport=udp>
Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,NOTIFY,REFER,MESSAGE
Content-Type: application/sdp
Content-Length: 520
Session-Expires: 3600;
Allow-Events: refer..
        v=0
        o=sip 2147483647 1 IN IP4 1.1.1.1
        s=sip
        c=IN IP4 1.1.1.1
        t=0 0
        m=audio 29784 RTP/AVP 8 0 4 18 18 18 18 96 3 98
        a=rtpmap:96 telephone-event/8000
        a=sendrecva=ptime:20
        a=rtpmap:18 G729AB/8000
        a=rtpmap:18 G729B/8000
        a=rtpmap:18 G729A/8000
        a=rtpmap:18 G729/8000
        a=rtpmap:4 G723
```

Figure 6. An INVITE from an attack trace

H. Results, alerts, and e-mail notification

We have two type of alerts:

- A message alert: contains all the conclusions that are inferred about a SIP message,
- A composed alert: contains information about a list of correlated SIP messages.

The alerts are provided to the command line interface and logged in text and HTML formats. The honeypot can be configured to send alerts to the administrator by e-mail. An example of a message alert is shown in Figure 5.

The honeypot source code is distributed under the GPLv3[10] license. The latest release is available at the project home page: http://artemisa.sourceforge.net/.

V. Experimentation and case studies

We extensively tested the honeypot performance in terms of robustness and accuracy. The goal of the robustness tests is to ensure that the honeypot does not easily crash when it is targeted by attacks or when it is abnormally flooded. The goal of the accuracy tests is to verify that it makes the good interpretation of received SIP messages.

In the sequel, we start by showing the offline interpretation of a SIP message detected by a deployed sensor. After that, we present our test-bed and summarize the results of 4 scenarios involving several attack tools. We show how responses can be triggered at parallel protection levels.

A. Interpretation of a SIP message

The INVITE message depicted in Figure 6 is reported by the Norwegian chapter (*http://www.honeynor.no/2009/09/20/citibank-uk-number-was-target-for-a-lawnmower-telephone-attack-today/*). We assume that this message is received by our honeypot and interpreted by the classifier decision tree. The following remarks are obtained:

- The User-Agent header is missing: We cannot estimate what tool has been used.
- Check DNS returns negative. The IP in the SDP (1.1.1.1) shows that the message is spoofed.
- Check if host is up and if SIP, media, or both ports are open: The host and ports are surely not available.
- Check for ACK and received media: No media delivery or ACK are noticed.

At this level, the received message is reported as spoofed: No real SIP state machine is involved at the attacker side. The used IP is spoofed or it belongs to a compromised machine. To get the overall picture, the other reported messages must be considered and correlated. If all the messages target the same extension, the correlator reports a flooding. If several extensions are targeted in a short period of time, three conclusions are possible:

1) The goal of the attacker may be making maximum disturbance by ringing all the phones in the domain. This is likely true if the attack persists.
2) The goal of the attacker is to enumerate the possible extensions in the domain. This is likely true if a large set of extensions is targeted (user part of the request URI). One can wonder, however, why the attacker does not use OPTIONS scanning which is more stealthy. In the same time, one cannot assume the logical behavior of the attacker.
3) The goal is to identify the gateways SIP/PSTN. This is true if a few messages are targeting several gateways (domain part of the request URI).

B. Testing against attack tools

Physically our test-bed is composed of two machines: The first machine contains the honeypot and an Asterisk[11] PBX

[10] http://www.gnu.org/licenses/gpl.html

[11] http://www.asterisk.org/

server. The second machine contains two bridged virtual machines: one representing the attacking tools and one having a soft-phone to emulate a legal external caller. We use a hard-phone to represent a user of the Asterisk server that has to be protected. The hardphone registers extension "305" at the Asterisk server. The honeypot registers extensions "300-304" and "306-310" in order to protect the targeted extension from both sides. Another option is to configure Asterisk to forward all the calls towards unregistered extensions to the honeypot. Next we give 4 case studies and we show how the honeypot can react and help to support the test-bed domain security.

1) Catch and crash SIPVicious: In this scenario, we launch the SIPVicious tool against our domain. The SIPVicious has three main scripts: (1) SVMAP that identifies hosts with open SIP ports, (2) SVWAR that identifies valid extensions in the domain and (3) SVCRACK that cracks weak authentication passwords using dictionary attacks. In addition, SVCRACK is a script given with the SIPVicious suite that can crash the attacker tool. We use this script at the honeypot side when the SIPVicious tool is notified. This can be shown in the following rule:

```
[rule_sip_vicious]
type = classification
context = INVITE
pre-action = assign %a Check_Fingerprint()
condition= %a == "SIPVicious"
action = system 'python /sipvicious/svcrash.py \
 -d SipMsg.Contact.Ip \
 -p SipMsg.Contact.Port'
```

In the future, we aim to discover vulnerabilities in the VoIP hacking tools by using "fuzzing" techniques[12]. The discovered vulnerabilities will be incorporated into the response scripts used by the honeypot to counter-attack the malicious sources.

2) Catch and block SPIT at the dial-plan level: In this scenario (Figure 7), we launch the Spitter tool [3] against our domain. When the honeypot detects the delivered SPIT calls, it adds the source to a blacklist in the Asterisk database (AstDb). The hardphone extension is protected by an AGI (Asterisk Gateway Interface) script within the Asterisk dial-plan. In other words every time this extension is called, the AGI script is run first. The AGI script checks if the call source is blacklisted before forwarding the call to the destination. In this way, the SPIT calls are blocked while the "good" calls are still routed. The blacklisting is based on the IP address for an external source and on the SIP URI for an internal (registered) source.

3) Catch and block a scanning at the IP level: In this scenario, we launch a scanning attempt against our domain. When the honeypot classifies the received message as scan, it adds a firewall rule at the IP firewall level (IPTables) in order to block the attack at an early stage.

4) Fingerprint a bunch of SIP attacking tools: We launch a bunch of SIP hacking tools against the honeypot in order to test its robustness. Particularly, the SIP parsing robustness is

[12]"Fuzzing is a method for finding bugs and vulnerabilities by creating different types of packets for the target protocol that push its specifications to the breaking point."[3]

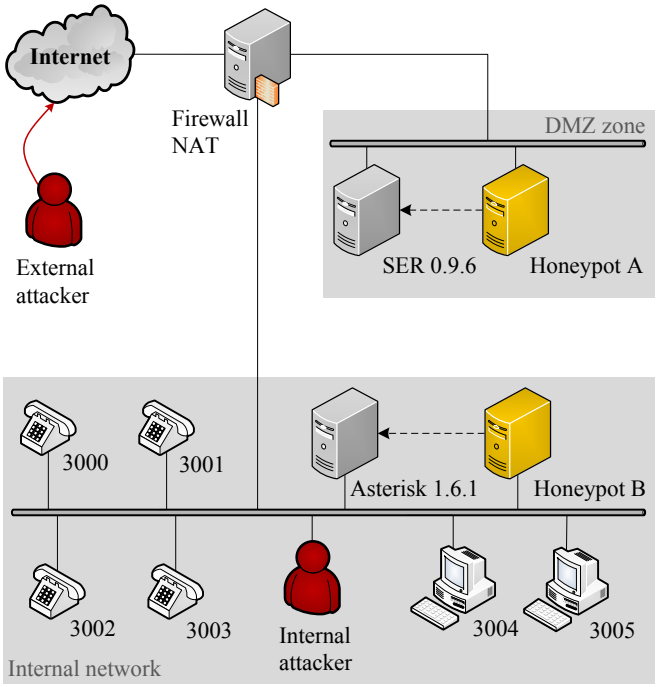

Figure 7. The test-bed

inherited from the PJSIP stack. In the same time we show the fingerprinting capability of the honeypot. The following tools are used:

- PROTOS Test-Suit (c07-sip): a SIP Torture Test that allows sending a major amount of malformed messages;
- Sipscan [3]: supports REGISTER, OPTIONS and INVITE scanning;
- SIPVicious: enumerates SIP servers in a given IP range by sending OPTIONS or INVITE messages;
- Inviteflood [3]: simple tool that allows several types of flooding based attacks;
- Sipp: allows testing the SIP servers performances under stress conditions;
- Sipsak: command-line SIP testing tool helpful for flooding and robustness tests;
- Spitter: works over Asterisk and automatically generates calls with an audio message to be delivered;
- SIP Send Fun[13]: supports several fuzzy INVITE messages for robustness testing;
- SipBot[14]: remotely controlled SIP attack tool supporting several attack commands.
- VoIPER[15]: a set of several tools for attacks like fuzzing and torturing.

The honeypot software shows good performance in terms of robustness: no crash is noticed even against the fuzzing and the flooding tools. It also demonstrates good performances in terms of the interpretation accuracy. This is the advantage of the rule-

[13]http://www.security-scans.de/
[14]http://gforge.inria.fr/projects/voipbot/
[15]http://voiper.sourceforge.net/

TABLE I
TESTING RESULTS

Tool	Fingerprint	Field	Real IP?	Field
Sipp	"Sipp"	From	Yes	From
Inviteflood	"Elite 1.0"	User-Agent	Yes	From
Spitter	"Asterisk"	From	Yes	From
SIPSCAN	"X-Lite"	User-Agent	Yes	Contact
SIPVicious	"friendly-scanner"	User-Agent	Yes	Via
Sipbot	"Twinkle"	User-Agent	Yes	From
SIP Send Fun	"Bad Guy Scripting Host"	User-Agent	No	-
PROTOS	'$\backslash d$ $\backslash d$ $\backslash d$ IN IP4' where $\backslash d$ increments	Owner (SDP)	No	-
Sipsak	"sipsak"	From	Yes	Contact
VoIPER	"VoIPER"	From	No	-

based approach when dealing with well-known attacks. We are able to identify and fingerprint all the attack tools. The results are summarized in Table I. The first two columns represent the regular expression and the fields used in the fingerprinting rules, respectively. The third column indicates if a real IP exists in the messages sent by the tool or if all the IPs used are spoofed. The last column indicates in which header or field the real IPs can be found.

VI. Conclusion and future works

We have introduced an innovative security monitoring complement to VoIP SIP-specific architectures by proposing an interactive VoIP-specific honeypot called Artemisa. The honeypot is able to contribute to the security of VoIP domains in different directions such as early stage detection, attack signatures collection, SPIT and scan mitigation. The honeypot invokes third-party reconnaissance tools in order to interpret the received messages and reports its finding in real time. The software has been extensively tested and demonstrated to be ready for incorporation in production domains. Also, to our knowledge, this is the first study on the fingerprinting and characterization of the existing VoIP "script-kiddies" tools. We aim by this contribution to encourage the deployment of VoIP honeypot sensors in order to build a security expertise in this field at an early stage.

In the future, we will provide a complete reference for Artemisa's rule description: syntax, rule types, attributes, operators, keywords and built-in operations. We will also study how to improve the processing performance of the honeypot when the size of the rule set increases.

Another goal is to develop domain honeypots that are more attractive in order to trap more sophisticated attackers. We want to innovate means to emulate high financial values such as gateway connections to the PSTN, premium-rate numbers or the telephony network of a prestigious business. Domain honeypots should regroup SIP infrastructures, PBX services, and individual user agent honeypots such as Artemisa.

We aim to incorporate the honeypot in state of the art VoIP intrusion detection systems such as SPIT prevention, advanced fingerprinting and anomaly-payload detection systems. We therefore have to define the interaction of the honeypot with the other security components in order to enable real-time and efficient adaptation of the security policy in the VoIP domains. In the longer term, a large deployment of the honeypot sensors will help to establish a black list of malicious spamming sources, attack tools signatures and a database of malicious SIP payloads.

Acknowledgments

We would like to thank the PJSIP project since we have used the PJSIP and the PJMEDIA libraries in our implementation. We also would like to thank Dr. Radu State from the University of Luxembourg for his advices. We thank Pablo Masri from Blas Pascal University for his contribution.

References

[1] I. Arce. Voices, I hear voices. *IEEE Security and Privacy*, 4(4):80–83, 2006.

[2] D. Butcher, X. Li, and J. Guo. Security challenge and defense in voip infrastructures. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 37(6):1152–1162, 2007.

[3] D. Endler and M. Collier. *Hacking Exposed VoIP: Voice Over IP Security Secrets & Solutions*. McGraw-Hill, Inc., New York, NY, USA, 2007.

[4] D. Geneiatakis, G. Kambourakis, T. Dagiuklas, C. Lambrinoudakis, and S. Gritzalis. SIP security mechanisms: A state-of-the-art review. In *Proceedings of the Fifth International Network Conference (INC 2005)*, pages 147–155, Samos, Greece, July 2005.

[5] D. R. Kuhn, T. J. Walsh, and S. Fries. Security Considerations for Voice Over IP Systems. National Institute of Standards and Technology (NIST), Special Publication, 800-58, January 2005.

[6] M. Nassar, R. State, and O. Festor. VoIP Honeypot Architecture. In *IM 2007*, pages 109–118, Munich, Germany, May 2007. IEEE Communications Society.

[7] V. M. Quinten, R. van de Meent, and A. Pras. Analysis of Techniques for Protection Against Spam over Internet Telephony . In *Proc. of 13th Open European Summer School EUNICE 2007*, July 2007.

[8] J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, and T. Ewald. Detecting SPIT calls by checking human communication patterns. In *IEEE International Conference on Communications (ICC 2007)*, June 2007.

[9] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Rfc3261: SIP: Session Initiation Protocol, 2002.

[10] R. Schlegel, S. Niccolini, S. Tartarelli, and M. Brunner. Spam over Internet Telephony (SPIT) Prevention Framework. In *Proc. of the IEEE GLOBECOM Conference 2006, San Francisco, USA*, November 2006.

[11] H. Sengar, R. Dantu, and D. Wijesekera. Securing VoIP and PSTN from integrated signaling network vulnerabilities. In *1st IEEE workshop on VoIP Management and Security (VoIP MaSe)*, Vancouver, Canada, April 2006.

[12] D. Sisalem, J. Floroiu, J. Kuthan, U. Abend, and H. Schulzrinne. *SIP security*. John Wiley & Sons, 2009.

[13] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Professional, 2002.

[14] T. J. Walsh and R. Kuhn. Challenges in securing Voice over IP. *IEEE Security and Privacy*, 3(3):44–49, 2005.